A New Architecture for Automated Modelling

Neil Smith*

Department of Computing and Information Sciences
De Montfort University, Milton Keynes, UK

Abstract

Existing automated modelling systems either rely on large, complex libraries or require complete access to the modelled system's behaviour, neither of which is desirable. To address these problems, a simpler architecture for modelling knowledge is described, based on the separation between ideal models of components and corrections that can be applied to these ideal models. The use of this architecture to develop accurate model boundaries is described, based on consideration of interactions within such ideal models. A novel algorithm for refining models is also proposed. This algorithm considers behavioural differences between models and applies the corrections that cause the greatest differences in behaviour. Finally, some models generated by this method are shown to be parsimonious.

Introduction

Existing automated modelling systems can be divided into two broad categories on the basis of how they develop models (Schut & Bredeweg, 1996). Model composition systems (*e.g.* Falkenhainer & Forbus (1991) and Iwasaki & Levy (1994)) are characterised by possessing a library of complex model fragments that are combined to form the model. This model composition process is controlled by applicability conditions in the model fragment library. In contrast, model induction systems (*e.g.* Addanki, Cremonini, & Penberthy (1991) and Amsterdam (1992)) have a very simple library structure, with model development occurring by comparing the behaviour of the model to that of the referent system.

These approaches have contrasting and complementary advantages and disadvantages. The libraries used in model composition systems are complex and difficult to develop (owing to the need to ensure that the applicability conditions are consistent). Such libraries are also restricted in flexibility: the modelling system is restricted to consider only those combinations of simplifications that are contained in the library. However, the structure of the modelling library is used to guide the development of the model boundary.

Model induction systems, on the other hand, use much simpler knowledge bases. As there are no separate submodels, there are no applicability conditions. Simplifying assumptions can be asserted and retracted independently. Model revision is performed by comparing the behaviour of the model to that of the referent system and using differences in behaviour to select the best alteration to make to the model. This approach is very flexible, but has the major constraint that the referent system's behaviour must be specified. In addition, model induction systems are incapable of determining a model boundary, but instead model everything in the referent system.

These considerations indicate a need to develop a modelling methodology that combines the benefits of the model composition and model induction approaches while eliminating their drawbacks. The structure of the modelling knowledge should be much simpler than compositional libraries while retaining sufficient sophistication to allow the modeller to identify the model boundary. The modeller should perform model revision based on the behaviour of the model, but this revision should be not be based on information beyond what is in a normal task definition.

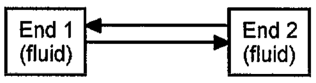
In this paper, we describe AIM, an automated modeller that has many of these features. We describe the architecture used in AIM to contain the modelling knowledge and show how this provides the power and flexibility desired. We then describe how AIM uses this architecture with a novel modelling algorithm to generate parsimonious models, and give some results showing this. Finally, we discuss some limitations of this approach.

Architecture of AIM

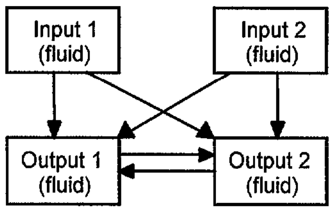
AIM is a component-centred modelling system, where discrete, separate components communicate with each other only through connections between specified ports. Knowledge in AIM is strictly partitioned between knowledge about physical *components* (stored in the component library) and knowledge about various models of these components (called *m-components*, and stored in the m-component library). Each of the libraries is organised into a hierarchy of frames, which allows AIM to infer default values for parameters that are not specified in the system description. The component library is used to store all the physical parameters relating to actual components, while the m-component library stores all the information relating to how these components can be represented in a model. A separately defined surjective function, with a domain including all components in the library, maps components to m-components. This function takes account of the state of the component and the type of interaction to which it is subjected.

M-components are similar to model fragments in that they are partial representations of a component's behaviour. Each m-component only specifies the component's

* Current Address CISMG, Cranfield University, Shrivenham, Swindon, SN6 7LA, UK. neil.smith@rmcs.cranfield ac uk



1*a*: Intra-actions in a **fluidPipe**



1*b*: Intra-actions in a sample **bathtub**

Figure 1: Intra-actions

response to interactions of a specified type when the component is in a specified state. For instance, the fluid flow properties of a steel pipe would be represented by a **fluidPipe** m-component; the same pipe's electrical properties would be represented by an **electricalConductor** m-component. A single m-component may represent many types of component: for instance, a **linearBlock** m-component can represent almost any free object subjected to a force.

The behaviour of an m-component is described by a fragment of a bond graph (Rosenberg & Karnopp, 1983) contained in the m-component. When the model is generated, these bond graph fragments are merged to produce a bond graph representation of the model; the bond graph is used to produce the state equations that yield the system's behaviour. The parameters that govern an m-component's behaviour (*e.g.* a **linearBlock**'s mass) are determined by the physical parameters of the component the m-component represents.

When an m-component is used in a model to represent a component, an effect (an interaction in a specified energy domain) at one port of a component will not necessarily cause similar effects at all other ports of the same component. To reflect this, each m-component has a set of intra-actions, which specify which of the m-component's other ports are affected by an effect at any one port. Intra-actions do not themselves indicate any causal direction within the m-component; they describe the possible causal orientations the m-component can support. Causal directions within an m-component are only defined when the m-component is placed in a model and the casual ordering of the whole model determined.

Intra-actions are normally symmetric, to reflect the symmetric nature of the relationships between effects in a component, *i.e.* the symmetric relationship between fluid flow rates at either end of a **fluidPipe** (figure 1*a*). However, the relationships between some effects are non-symmetric and the m-component's intra-actions reflect this. For example, the water flow rate from a tap into a bath affects the flow rate from the plughole, but the converse is not true (figure 1*b*). This information is used by AIM when it determines the model boundary; this is described in the next section.

M-components differ from model fragments in that an m-component only represents the *ideal* model of a component, *i.e.* a model in which all simplifying assumptions have been made. However, not all of these assumptions are valid in all circumstances: the viscosity of fluid in a pipe will be negligible if the fluid flows slowly and the model is only used to represent a small time scale. In other situations, the viscosity might be significant, and the basic **fluidPipe** m-component will need to be augmented to include the effects of fluid viscosity. This is achieved in AIM through the mechanism of corrections, which can be added to m-components to reflect the retraction of these simplifying assumptions. All corrections in AIM are examples of fitting approximations (Weld, 1992).

Corrections have a very similar structure to m-components. Each correction has at least one port, the attachment port, that controls how the correction is added to the m-component (though note that corrections attach to an m-component's body, rather than one of its ports). Some corrections, *e.g.* heating in an electrical resistor, introduce additional ports and intra-actions to the m-component to which they attach. The inclusion of such corrections can expand the model boundary, but a detailed examination of such corrections is outside the scope of this paper. Each correction contains a bond graph fragment that is merged into the bond graph of the m-component to which it applies. Corrections have various parameters that are defined by the physical parameters of the component to which they relate.

As each correction represents the retraction of a simplifying assumption, a correction can only be applied once to a particular instantiated m-component. However, the same correction can be applied to an arbitrary number of different instantiations of the same m-component, and can even be applied to different types of m-component. The application of corrections to one m-component is independent of the application of that correction to any other m-component in the model. The *correction candidates* of a model are all those corrections that can validly be added to the model, *i.e.* are not already present in the model.

This architecture for the modelling knowledge used in AIM allows for the combination of the benefits of the model composition and model induction approaches. The component-centred approach to modelling, combined with the m-components' intra-actions, provides the sophistication required to accurately determine the model boundary in response to a specified task definition. The separation of the corrections from the m-components allows AIM the freedom to select only the simplifying assumptions that are appropriate in this model.

Generating models

AIM requires three inputs before modelling can begin: a system description, a task definition, and a significance

```
function find_parsimonious_model ($sd$, $voi$, $cd$, $\varepsilon$)
% $sd$: system description
% $voi$: variables of interest
% $cd$: causal direction specified in task
% $\varepsilon$: significance threshold value
% $m$, $m'$, $m_{next}$: models
% $\mathsf{R}(m)$: correction candidates of $m$
% $r$: correction

begin
  $m$ = coherent (expand_boundary ($sd$, $\varnothing$, $voi$, $cd$))
  repeat
    $\mathsf{R}(m)$ = all valid correction candidates for $m$
    $J_{max} = 0$
    for each $r \in \mathsf{R}(m)$
      $m'$ = coherent (expand_boundary ($sd$, $m + r$,
            intra-actions($r$, attach_port, $cd$), $cd$))
      $J = \vartheta(m, m')$    % difference in behaviour
      if $J > J_{max}$,
        $J_{max} = J$
        $m_{next} = m'$
      end
    end
    if $J_{max} \not\ll \lambda$    % $J_{max}$ is significant
      $m = m_{next}$
    end
  until $J_{max} \ll \lambda \vee \mathsf{R}(m) = \{\}$
  if $\mathsf{R}(m) = \{\}$
    return nil    % cannot guarantee an adequate model
  else
    return $m$
  end
end
```

Figure 2: AIM's Algorithm

threshold value. The system description is given in terms of the components that make up the system and the connections between them. The task definition defines both the variables of interest and a causal direction. The causal direction indicates whether the model created should either describe the effects these variables have on the system, or describe what factors in the system affect these variables. This information is used in the model boundary analysis. The significance threshold value defines when AIM should regard two models' behaviours as significantly different.

AIM generates a single output: the model. The model consists of a 3-tuple of `(m-components, connections, corrections)`, where `m-components` is a set of instantiated m-components; `connections` is a set of connections, each connection consisting of a domain of interaction and a set of ports at that connection; and `corrections` is a set of instantiated corrections. If a component is subjected to many effects, it may be represented in the model by several m-components.

Models are developed in AIM in a two-stage process. The first stage determines the model boundary, which identifies the physical and behavioural extent of the model. The model boundary defines the initial model. AIM then moves on to test all the simplifying assumptions made in the initial, idealised model and, by adding corrections, retracts those assumptions that are unjustified. The process repeats until AIM determines that all the remaining simplifying assumptions are justified; when this occurs, modelling stops. This algorithm is shown in figure 2. Models are generated for only a single operating region: different operating regions of a system will require different models. In addition, as AIM is purely a model builder, AIM cannot control the transitions between distinct models.

The discussion below of AIM's algorithm will be illustrated by showing how AIM generates a model of a water-filled syringe (figure 3). The model is intended to show all the effects of a force applied by the finger.

Finding the model boundary

The determination of the model boundary focuses on the ports in the model and what other ports in the system are either affected by these ports, or are needed to explain effects at these ports. As additional ports are identified, the model boundary expands to include the m-components to which these ports belong. The algorithm, shown in figure 4, centres on the *boundary analysis queue*. This stores details of all the ports in the model (with their corresponding effects) that are currently on the model boundary. However, there can be no ports on a correctly-drawn model boundary as any ports on the boundary represent an under-specified effect: each port forms part of a junction, and the effects at a junction depend on all the ports at that junction. For example, the consideration of the current on one lead at an electrical junction requires the consideration of the electrical properties of all the other leads at that junction. Therefore, the boundary analysis continues until the boundary analysis queue becomes empty.

The boundary identification process starts by identifying the ports and effects referred to in the task definition and placing these in the boundary analysis queue. In the example of the syringe, the boundary analysis queue will initially contain the single port/effect combination `(finger, end, linearMechanical)`.

Boundary expansion performed port by port, as shown in figure 4. When several ports are connected, an effect at one port requires AIM to consider the same effect at all the connected ports (providing the connection supports the inter-action). All the connected ports are added to the boundary analysis queue and the connection is added to the model. For instance, because the finger touches the end of the plunger handle, this latter port will be added to the boundary analysis queue. However, if a port already exists in the model as part of a connection, the connection must already have been processed and AIM does not process this connection again.

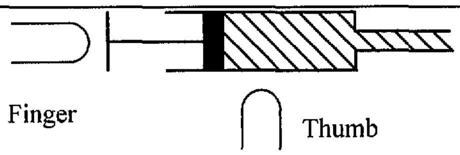


Figure 3: The Syringe

```
function expand_boundary (sd, initmodel, baq, cd)
% m: model
% bap: boundary analysis port
% p: port
% cp, mp: additional ports
% mc : m-component

begin
  m = initmodel
  while $baq \neq \{\}$
    bap = dequeue (baq)
    if $bap \notin m$
      cp = system ports connected to bap
      enqueue (baq, cp)
      m = m + cp
    end
    mc = correct_m-component (bap, sd)
    if $mc \notin m \vee \neg(\exists p$ such that $[p \in m \wedge$
        $p \in$ intra-actions $(mc, bap, \neg cd)]$
      mp = intra-actions (mc, bap, cd)
      enqueue (baq, mp)
      m = m + mc
    end
  end
  return m
end
```

Figure 4: Algorithm for finding the model boundary

AIM also needs to determine the correct m-component to use to represent each component in the model. The component is identified directly from the port description. The combination of port description, effect, and component type and state (found from the system description) is sufficient to specify the correct m-component to use to represent that component.

AIM checks whether this port requires further processing by examining the m-component's intra-actions. If the current port was placed in the boundary analysis queue through the analysis of other ports of this m-component (*i.e.* if there exists in the model a port of the current m-component whose consideration would have caused this port to be included), AIM can move on to the next port in the boundary analysis queue. If not, details of the m-component, and the component to which it relates, are added to the model.

The m-component's intra-actions are then used to determine which of the component's other ports have effects that are significant in the model. The causal information in the task description indicates whether to use the intra-actions entering the port (to find the ports that affect the port in question) or the intra-actions leaving the port (to find the ports affected by this port). This only becomes significant where an m-component's intra-actions are not symmetric. All ports mentioned in the relevant intra-actions are added to the boundary analysis queue. In the syringe, the finger is modelled as an exogenous application of force, and so has no intra-actions. The plunger handle, modelled as a rigid bar, has a symmetric intra-action that relates mechanical effects at the ends to each other: the finger pressing at one end of the handle prompts AIM to consider the effects of the other end of the handle on the plunger.

The components found to be inside the model boundary in the syringe example, together with their m-components, are shown in table 1. Note that all the components are represented by ideal models, and note that the atmosphere surrounding the syringe is explicitly included in the model boundary.

Refining the model

The initial model, found during the identification of the model boundary, is likely not to be sufficient to address the task specified. The model is made adequate by the addition of corrections. Corrections can be added for two reasons: to ensure coherence in the model, or to reduce the behavioural difference between the model and the referent system.

A model is said to be *incoherent* if it does not yield a complete set of state equations[1]; this normally reflects a physically impossible situation in the model. The initial model of the syringe, shown in table 1, would predict an infinite velocity for the ram, due to the lack of friction or other similar phenomena in the model. Such errors in the model are removed by the addition of corrections. The function **coherent**, described by Smith (1998), identifies the corrections that could potentially eliminate the error and selects the one that has the greatest effect on the model's behaviour. In the syringe example, this correction is the friction between the plunger and the cylinder.

However, the main reason for including corrections in a model is to reduce the behavioural difference between the model and the referent system. We take the view that models are always intended, at some level, to explain the behaviour of the referent system. This allows us to define a model as adequate for a task if the behaviour of the model is not significantly different from the behaviour of interest of the referent system.

If we assume that AIM's libraries contain all possible corrections (thus ignoring any closed world assumption), the behaviour of a model can be brought as close as desired to that of the referent system by the inclusion of corrections in the model. Normally, only a few of the possible corrections will have significant effects on the model's behaviour; the objective of modelling is to identify which corrections fall into this category. A model that contains all the significant corrections will be adequate for the specified task. An adequate model that contains no other corrections is parsimonious.

The problem is how to assess the effect of each correction candidate on the behavioural difference between the model and the referent system. Model induction systems do this by generating the behaviour of the model and com-

[1] State equations are found from the bond graph representation of the model. This reliance on bond graphs restricts the level of granularity of models.

Component	Modelled as
Finger	Source of force
Thumb	Not modelled
Ram	Rigid, massless bar
Plunger	Watertight, rigid, frictionless plunger
Cylinder	Rigid container of inviscid fluid
Nozzle	Rigid container of inviscid fluid
Atmosphere	Source of (zero) pressure

Table 1: M-components of the Syringe

paring it to a trace of the referent system's behaviour, provided as part of the problem specification. This requires that the referent system's behaviour be known.

Alternatively, AIM could produce a "most complex" model to produce a behaviour trace against which other models are compared. Unfortunately, this model would be formed under an arbitrary closed world assumption. Additionally, a model that contains all the corrections available under this closed world assumption might not be coherent. To make it coherent, AIM would have to arbitrarily eliminate corrections to ensure the model's coherence. Finally, increasing the knowledge in the modeller would increase the complexity of the initial model. If the additional complexity is not needed in the parsimonious model, including it only serves to increase the cost of modelling.

Instead of the above approaches, AIM takes the novel step of using the behaviour of an existing model as the base against which refinements are compared. When a model is generated, its behaviour is found. When each correction candidate is assessed, the model is revised to include this correction and the behaviours of the base model and the revised model are compared. This revision and comparison is repeated for all the model's correction candidates. If no correction causes a significant change in the model's behaviour, the model is deemed adequate and modelling stops. If the largest change in behaviour is significant, then the correction that caused that change is included in the model. This revised model becomes the current model and the process repeats. This approach relies on the effects of corrections on the model's behaviour being nearly independent, and on the insignificant correction assumption (see below).

AIM's search strategy through the space of possible models is similar to a steepest ascent hill climbing search. If the effects of corrections on the model's behaviour were truly independent, the order of inclusion of the significant corrections would be irrelevant. However, while corrections' effects are nearly independent, they are not truly independent. This has the result that if AIM is given a choice between two correction candidates, both of which have a significant effect on the model's behaviour but with one having a larger effect than the other, AIM should first include the correction candidate with the largest effect and reassess the model.

Whichever correction candidate is chosen, there is a chance that the remaining correction candidate will not have a significant effect on the modified model. If the corrections' effects are nearly independent, a correction candidate with a large effect will continue to have a large, significant effect, ensuring that this correction is included in a later refinement of the model. However, the small effect of the other correction candidate might become insignificant in a later model, meaning that this correction might not be included. In this case, this correction is not necessary to form an adequate model, and should not be included in the final, parsimonious model. This consideration leads to AIM's steepest ascent hill climbing search strategy. By including early the corrections with the largest effects on behaviour, AIM defers decisions on correction candidates with smaller effects until their impact becomes clearer.

In AIM's current implementation, behaviours are generated by simple numerical methods and compared using an extension of the integral-absolute error performance index (Palm, 1983). However, any method of generating and comparing behaviours will suffice for AIM's algorithm to work so long as behavioural differences can be found, ordered, and tested for significance.

Rather than compare the behaviours of all variables in the model, AIM selects a subset of variables to track, depending on the structure of the model and the causal direction in the task definition (if the model is to explain how the specified variables are affected, only these variables are tracked; if the effect of the specified variables on the rest of the model is to be found, all state variables and outputs are tracked). If there are n tracked variables in the base model, of which $v_i(t)$ is the ith tracked variable in the base model and $v_i'(t)$ is the corresponding variable in the modified model, the difference in behaviour over the period $[0, \tau]$ is given by:

$$\vartheta(m, m') = \max_{1 \le i \le n} \left(\frac{\int_0^\tau | v_i'(t) - v_i(t) | \, dt}{\int_0^\tau | v_i(t) | \, dt} \right)$$

where the performance index is normalised to give a measure of the relative difference in behaviour.

To show how AIM includes these corrections in the model, consider the simplest coherent model of the syringe (the model shown in table 1, with the addition of the plunger friction correction to ensure model coherence). If the cylinder is made of steel, the remaining correction with the largest effect is the inertia of the water in the nozzle: including this correction gives a performance index of 0.01218. Given a significance threshold value of 0.1, this is an insignificant effect. Therefore, the simplest model, excluding this correction, is deemed adequate. However, if the cylinder is made from soft rubber, the deformation of the cylinder is has the largest effect of all the corrections with a significant performance index of 0.2186. Following the algorithm in figure 2, this means that this correction is added to the model and refinement must continue.

The Insignificant Correction Assumption

AIM uses the insignificant correction assumption to determine when modelling is to stop. Modelling stops when the current model, m, is adequate. m is adequate iff the performance index between m and the referent system S is insignificant, *i.e.* $\vartheta(m, \mathsf{S}) \ll 1$ ($x \ll 1 \leftrightarrow x < \varepsilon$, where ε is the significance threshold value).

Turning attention to the mythical most complex model m_∞ (for which $\vartheta(m_\infty, \mathsf{S}) \equiv 0$), there are some corrections r in m_∞ for which $\vartheta(m_\infty - r, m_\infty) \ll 1$. Such corrections are termed *insignificant*, as they have no significant effect on the behaviour of the most complex model. $\mathtt{I}$ is the set of all such insignificant corrections:

$$\mathtt{I} = \{r \in \mathsf{R} : \vartheta(m_\infty - r, m_\infty) \ll 1\}$$

where R is the set of all corrections.

If we assume that the effects of all corrections on the behaviour of the model are independent, it is true that:

$$\vartheta(m_\infty - \mathtt{I}, m_\infty) \ll 1$$

which means that $m_\infty - \mathtt{I}$ is an adequate model.

The insignificant correction assumption states that no insignificant correction has a significant effect on the behaviour of any model:

$$\forall r \in \mathtt{I}, \vartheta(m, m + r) \ll 1$$

Therefore, given $\mathsf{R}(m)$ is the set of valid correction candidates for m:

$$\begin{aligned} \forall r \in \mathsf{R}(m), \vartheta(m, m + r) \ll 1 &\rightarrow \mathsf{R}(m) \subseteq \mathtt{I} \\ &\leftrightarrow m \supseteq m_\infty - \mathtt{I} \\ &\rightarrow \vartheta(m, m_\infty) \ll 1 \\ &\leftrightarrow \vartheta(m, \mathsf{S}) \ll 1 \end{aligned}$$

This allows AIM to detect an adequate model by examining the effects of the remaining corrections on the behaviour of that model. If no correction candidate has a significant effect on the behaviour of the model, all the correction candidates must be insignificant; therefore, the model is adequate.

However, the assumption that all corrections have totally independent effects on the model's behaviour is not wholly true. In linear systems, corrections will generally have independent effects, but in order to allow for synergistic effects between corrections, the independence relation is weakened to:

$$\vartheta(m_\infty - \mathtt{I}, m_\infty) \ll 1/\lambda$$

which means that m is adequate when:

$$\forall r \in \mathsf{R}(m), \vartheta(m, m + r) \ll \lambda$$

Cylinder	Force	Corrections
Steel	Constant	Plunger friction*
Steel	Varying	Plunger friction* Nozzle fluid inertia Plunger leakage
Rubber	Constant	Plunger friction* Cylinder deformation Nozzle fluid drag* Nozzle fluid inertia
Rubber	Varying	Plunger friction* Cylinder deformation Nozzle fluid drag* Nozzle fluid inertia

Table 2: Corrections added to the Syringe
* Correction added to ensure model coherence.

The value to use for λ seems to be problem dependent, but an investigation of the combined effects of corrections (Smith, 1998) suggests that using $\lambda = ½$ is reasonable.

Results

AIM's libraries contain approximately 40 components, 40 m-components, and 20 corrections. AIM has been used to generate models for several systems, including a cascaded tank system, a heat exchanger, and a simple tachometer, each containing about a dozen components. Each system was modelled under a variety of different conditions; the results are described by Smith (1998). However, only the syringe system (figure 3) is discussed here.

In order to accurately describe the effect of the force applied by the finger, the basic model of the syringe (table 1) is augmented with various corrections; these are shown in table 2. However, if the physical parameters in the system description are altered, different corrections will be appropriate in different circumstances. Table 2 shows how the necessary corrections change depending on whether the finger exerts a constant or rapidly varying force, and whether the syringe cylinder is made from steel or soft rubber. The corrections are shown in the order in which they were added to the model (*i.e.* the most significant correction first). All models were built under the same significance threshold value (ε) of 0.1.

To determine whether these models are adequate, a very complex model was built manually and the behaviours of the generated models were compared to this complex model. The results are shown in the third column of table 3. In addition, the precursors of these models were also compared to the complex model and the results of these comparisons can be found in the fourth column of table 3.

As can be seen from these results, the models produced

Cylinder	Finger Force	$\vartheta(m, m_\infty)$	$\vartheta(m_-, m_\infty)$
Steel	Constant	0.089516	—
Steel	Varying	0.023991	0.451486
Rubber	Constant	0.035248	0.728074
Rubber	Varying	0.031973	1.01271

Table 3: Performance Indices for Syringe Models

by AIM are adequate ($\vartheta(m, m_\infty) \ll 1$, *i.e.* $\vartheta(m, m_\infty) < \varepsilon$) while the precursor models are not ($\vartheta(m_-, m_\infty) \not< 0.1$). This means that AIM produced the simplest adequate, *i.e.* parsimonious, models.

Related Work

Nayak & Joskowicz (1996) describe a typical model composition system, whose modelling knowledge base shares many features with AIM. Model fragments are organised into hierarchies to allow the reuse of modelling knowledge, and articulation rules (similar to AIM's intra-actions) are used to infer what effects are induced in a component in response to a given effect. However, the modelling algorithm is entirely reliant on the structure of the modelling library to guide the modelling process. This is only achieved at the cost of embedding modelling assumptions throughout the library. This is shown in the different but overlapping hierarchies of model fragments, and in the arbitrary nature of the structural and behavioural preconditions governing when model fragments can be used. Smith (1998a) shows how most of these embedded assumptions can be avoided.

In contrast, MM (Amsterdam, 1992) is a typical model induction system that depends wholly on behavioural considerations. However, MM compares qualitative behaviours, which greatly reduces MM's ability to resolve qualitatively similar but quantitatively different behaviours. AIM's quantitative method of behavioural comparison allows it to produce models more appropriate to the specified task. In addition, while MM is able to correctly identify the different behaviourally significant regions within the system (the lumping problem), little attention is paid to the determination of the model boundary.

DME (Iwasaki & Levy, 1994) is a model composition system that uses relevance reasoning to guide modelling. Each model fragment contains a set of modelling assumptions under which it is valid. As the model boundary expands, these assumptions are asserted and retracted, which can prompt the modeller to revise earlier decisions. DME's performance depends critically on the correctness of these assumptions. Iwasaki & Levy do not guarantee this is the case, and instead offer the library coherence assumption. AIM's simpler knowledge base structure obviates the need to make any such coherence assumptions when developing the libraries, and AIM ensures the coherence of models as they are being built.

Williams & Raiman (1994) have produced Charicatures, a radically different modelling system based on the simplification of the equations that represent the most complex model. Their major contribution is their exploration of the concept of a model's domain of validity, which describes the situations in which the model can be used with confidence. This is used to indicate when model transitions are required. However, focusing solely on the equations is a very shallow approach as it ignores the physics that underlies the algebraic formulation. This restricts the application of Charicatures to situations that have already been modelled as algebraic systems.

Conclusions

We have described AIM, an automated modelling system that uses a novel architecture to provide both power and flexibility during the modelling process. AIM implements an algorithm that does not rely on external sources of information or fine structure in the modelling knowledge to guide and halt modelling. Instead, AIM compares the behaviours of successive models, including corrections that cause significant changes in the model's behaviour. We have shown that this approach to modelling usually generates parsimonious models.

However, this approach relies on the insignificant correction assumption and assumptions about the independence of corrections. Tests on other systems have shown that, occasionally, these assumptions are not sophisticated enough to always ensure parsimonious models. In addition, AIM's reliance on fitting approximations restricts the types of corrections that can be made.

References

Addanki, S., Cremonini, R., Penberthy, J. S. (1991), Graphs of Models, *Artificial Intelligence* **51**:145–177.

Amsterdam, J. (1992), Automated Modeling of Physical Systems, in Falkenhainer, B & Stein, J. L. (eds.), *Automated Modelling, DCS* **41**, American Society of Mechanical Engineers, pp. 21–30.

Falkenhainer, B., Forbus, K. (1991), Compositional Modeling: Finding the Right Model for the Job, *Artificial Intelligence* **51**:95–143.

Iwasaki, Y., Levy, A. Y. (1994), Automated Model Selection for Simulation, *Proceedings AAAI-94*, pp. 1183–1190.

Nayak, P. P., Joskowicz, L. (1996), Efficient Compositional Modeling for Generating Causal Explanations, *Artificial Intelligence* **83**:193–227.

Palm, W. J. (1983), *Modeling, Analysis, and Control of Dynamic Systems*, New York: J. Wiley & Sons.

Rosenberg, R. C., Karnopp, D. C. (1983), *Introduction to Physical System Dynamics*, New York: McGraw-Hill.

Schut, C., Bredeweg, B. (1996), An Overview of Approaches to Qualitative Model Construction, *Knowledge Engineering Review* **11**(1):1–25.

Smith, N. (1998), Reducing the Need for Assumptions in the Automated Modelling of Physical Systems, PhD thesis, School of Computing, De Montfort University.

Smith, N. (1998a), Handling Assumptions in Automated Modelling, *Working Notes of the 12th International Workshop on Qualitative Reasoning.*

Weld, D. S. (1992), Reasoning about Model Accuracy, *Artificial Intelligence* **56**:225–300.

Williams, B. C., Raiman, O. (1994), Decompositional Modeling Through Charicatural Reasoning, *Proceedings AAAI-94*, pp. 1199–1204.